ESC-TR-2006-080

Technical Report
1121

Making Network Intrusion Detection Work with IPsec

C.D. McLain
A. Studer
R.P. Lippmann

11 May 2007

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

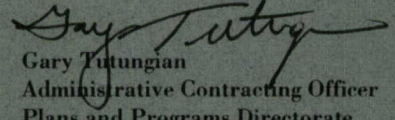
Prepared for the Department of the Air Force under Contract FA8721-05-C-0002.

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Department of the Air Force, ESC, under Air Force Contract FA8721-05-C-0002.

The ESC Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



Gary Tutungian
Administrative Contracting Officer
Plans and Programs Directorate
Contracted Support Management

Massachusetts Institute of Technology
Lincoln Laboratory

Making Network Intrusion Detection Work with IPsec

C.D. McLain
A. Studer
R.P. Lippmann
Group 62

Technical Report 1121

11 May 2007

Lexington Massachusetts

ABSTRACT

Network-based intrusion detection systems (NIDSs) are one component of a comprehensive network security solution. The use of IPsec, which encrypts network traffic, renders network intrusion detection virtually useless unless traffic is decrypted at network gateways. One alternative to NIDSs, host-based intrusion detection systems (HIDSs), provides some of the functionality of NIDSs but with limitations. HIDSs cannot perform a network-wide analysis and can be subverted if a host is compromised. We propose an approach to intrusion detection that combines HIDS, NIDS, and a version of IPsec that encrypts the header and the body of IP packets separately. We refer to the latter generically as Two-Key IPsec. We show that all of the network events currently detectable by the Snort NIDS on unencrypted network traffic are also detectable on encrypted network traffic using this approach. The NIDS detects network-level events that HIDSs have trouble detecting and HIDSs detect application-level events that can't be detected by the NIDS.

ACKNOWLEDGMENTS

This work was sponsored by David Kenyon of the Air Force Electronic Systems Command, GIG Network Architecture Office.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Computer attacks and misuse result in significant costs in downtime and lost or stolen information. To prevent or reduce the impact of these undesirable events, administrators often set up an intrusion detection system (IDS) to monitor activities and generate alerts or actions. IDSs use two methods to detect suspicious behavior: pattern matching and anomaly detection. Pattern matching IDSs use signatures or rules that describe undesirable events and perform some action when the pattern matches an event or data. They are best at detecting attacks which have a known signature. Anomaly detection IDSs look for behavior that deviates from "normal" behavior. Normal behavior is defined by selected characteristics of a baseline data set on which the anomaly detection system has been trained. Anomaly detection IDSs are designed to detect general system misuse and attacks for which no signature exists, often called zero-day attacks.

Pattern matching and signature detection IDSs are commonly used to detect and respond to misuse. Such systems, however, cannot detect that a third party is intercepting messages, or protect traffic after it exits the local network. To prevent eavesdropping and tampering at the network level, IPsec (Internet Protocol Security) [14] is used to encrypt and authenticate traffic as it flows across a network. The IPsec Encapsulating Security Payload (ESP) [13] protocol with authentication prevents unauthorized parties from reading or modifying the contents of packets.

IPsec can be used in either of two modes: transport or tunnel. With transport mode IPsec, the IP payload is encrypted and the IP header is left unencrypted. If transport mode IPsec is used, other parties on the network can see the source and destination addresses of packets but no other information. With tunnel mode IPsec, the entire packet is encrypted and a new IP header is added to the packet. If tunnel mode IPsec is used, the original source and destination addresses of the packet are hidden as well, providing network monitors with even less information. The secrecy provided by encrypting traffic with IPsec acts as a dual-edged sword. Malicious parties can no longer eavesdrop on network traffic. However, encryption hides the majority of traffic content from any intrusion detection system monitoring traffic in the network.

Administrators can install IDSs on the end hosts and use a Host-based Intrusion Detection System (HIDS). However, HIDSs are limited. Even with collaboration among HIDSs, HIDSs can miss network-level events, such as scans of unassigned IP addresses. In addition, a successful attacker can disable a HIDS. A Network-based Intrusion Detection System (NIDS) monitors traffic and detects suspicious events. NIDSs detect network-level events, but require more than the source and destination addresses of a packet to detect events and they often examine packet contents. This information is hidden when traffic is encrypted with IPsec.

Several researchers have proposed selective encryption schemes for providing network-based services, such as packet-filtering firewalls, access to data encrypted by IPsec [10, 12, 31]. In this report, we investigate the efficacy of using selective encryption with IPsec combined with HIDS to restore the functionality of Snort [19], a widely used NIDS. This approach uses selective encryption on an IP packet by breaking the packet into multiple encryption zones. This technique was proposed in 1999, largely for improving network performance, independently as Multilayer IPsec (ML-IPsec [31]) and Layered Encryption Security (LES [10]). Each encryption zone is assigned a

cryptographic key and a portion of the IP packet over which that key is used for encryption and decryption. Selective encryption enables a well-designed NIDS to detect network-wide events while HIDSs detect more application-specific events.

For this investigation, we restrict the number of encryption zones to two: one for the header information at the beginning of the packet and the other for the remainder of the packet, containing the application data. End hosts have access to the entire packet. The network services, Snort in this case, have access to the header information only. In the remainder of the report, this two-encryption-zone strategy will be referred to as "Two-Key IPsec," since we are interested in the impact of using this encryption strategy independent of a particular implementation.

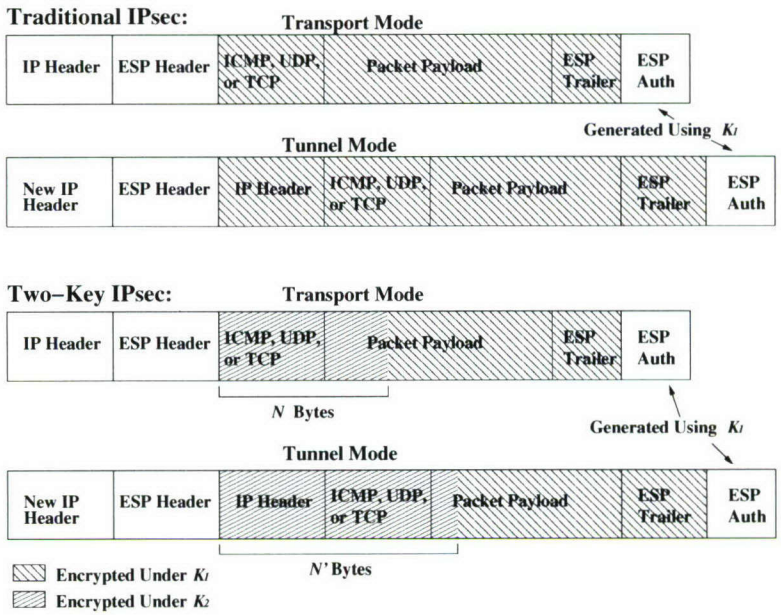


Figure 1. Traditional and Two-Key IPsec in Transport & Tunnel Modes

Figure 1 illustrates the difference between traditional and Two-Key IPsec. In the top two packet diagrams, the hashed area represents the data that traditional IPsec encrypts using a single key, K_1. Only the endpoints of the connection know K_1. In the bottom two packet diagrams of Figure 1, the two hashed areas show the effect of using Two-Key IPsec to encrypt the bytes of the packet with two separate keys, K_1 and K_2. The first N bytes of the ESP payload are encrypted with the second key, K_2. Note that for IPsec tunnel mode, N' bytes are encrypted using K_2, where $N' = N + L_{IP\_Hdr}$ and L_{IP_Hdr} is the length of the encapsulated IP header. K_2 is shared among the end hosts and the network-based service. K_1 is used to encrypt the remainder of the ESP payload and, as with traditional IPsec, is known only to the endpoints. In both cases, IPsec adds the ESP header and appends an integrity check value, or authentication data (ESP Auth). For simplicity, we show the authentication data as being generated for the encrypted portion of the packet using K_1. Using any key other than K_2 to generate the authentication data prevents

a network-based service from successfully modifying the contents of the packet, effectively giving any machine having only K_2 read-only access.

If the network-based service is trusted and assumed to be perfectly secure, the end hosts might as well encrypt the entire packet under a single key that is shared with the network service. However, Two-Key IPsec follows the principle of least privilege and insures that network services have access only to necessary information.

This work analyzes Snort to determine which rules require access to only the first bytes in each packet. We find that Snort requires access to entire packets to detect the majority of attacks. However, Snort does detect a crucial set of suspicious events by examining the first few packet bytes. More important, the limited types of events Snort can detect when Two-Key IPsec is used are those missed by HIDSs. In order to secure both network traffic and systems, we propose the deployment of NIDSs and HIDSs in conjunction with Two-Key IPsec.

The remainder of this report is organized as follows. In Section 2 we discuss work related to intrusion detection and encryption of network traffic. Section 3 shows how Snort examines packets and how we determine which bytes in the packets are required by different Snort rules. Section 4 introduces a taxonomy of the different types of events that Snort detects. In Section 5, we use this taxonomy to establish meaningful metrics of Snort's capabilities when different fractions of bytes in packets are accessible. Section 6 describes how HIDSs, a NIDS, and Two-Key IPsec can be combined to detect all suspicious events detectable by Snort. We discuss some issues with Two-Key IPsec in Section 7 and make concluding remarks in Section 8.

2. RELATED WORK

It is widely recognized that the use of IPsec can cause problems for network services such as firewalls, Performance Enhancing Proxies, and Network Address Translation, as well as NIDSs. A variety of solutions have been proposed involving decryption, signaling protocols, and packet modifications [17]. Several commercial products [3, 9, 28] co-locate an IPsec endpoint and NIDS at a network gateway. This solution allows the IDS to operate at the cost of leaving traffic in the clear within a local network. An attacker that has subverted a host on such a local network has access to all of the traffic destined for the network, not just traffic destined for the subverted host. Our work analyzes the usefulness of a NIDS when a portion of the IPsec-encrypted packet is made available to the NIDS, allowing the traffic to remain encrypted on a local network.

Other alternatives have been proposed when the IPsec endpoints are at the hosts and traffic is encrypted throughout the network. In one commercial solution [30], each host has an IDS that reports to a centralized server. While able to protect individual hosts against attacks, this approach is not able to detect scans of invalid addresses or perform more than rudimentary packet-filtering at the network firewall when the packets are encrypted in the network. There may also be a delay before the centralized server detects subversion of a host. Another solution provides third parties with a master key that can be used to examine the full contents of all packets [4, 5]. Although we leverage the idea of the master key, we feel that allowing access to the full packet contents would unnecessarily provide too much information to network devices.

Our work uses Two-Key IPsec to provide a NIDS access to only required data in each packet. Researchers have suggested using IPsec with multiple keys [10, 11, 31, 32] or with more data in the clear [1, 12] to support services on devices that are not at the endpoints of communication. The focus of those publications, and derivative works [20, 21], was largely to improve performance of flows, not to increase the security of the system. When security is discussed, it is usually in the context of packet-filtering firewalls. We examine the feasibility of using Two-Key IPsec by examining the full Snort rule set.

Later in this work we argue that both network-based and host-based intrusion detection systems are necessary to detect all types of attacks. This requirement is not a limitation of Snort, but reflects the complementary nature of NIDSs and HIDSs [2, 18, 24, 27, 29]. A NIDS can be better secured against compromise than can a HIDS. Thus, a NIDS is valuable for monitoring the health of all the hosts within its purview. A NIDS can also handle certain network-level events better than can a HIDS. Likewise, a HIDS can detect certain events, such as exploitation of applications, better than a NIDS. Our analysis of the Snort rule set confirms this complementary nature of NIDSs and HIDSs. In the case where both IPsec and intrusion detection are needed, we argue administrators should use Two-Key IPsec to protect the traffic on the network in addition to protecting the hosts.

3. HOW SNORT RULES EXAMINE PACKETS

To understand what portion of the packet Snort examines, we need to understand how Snort rules dictate what the Snort engine looks for when trying to detect suspicious traffic. Almost all of the rules examine the header of the packet to determine the source and destination addresses and ports. Snort rules can be divided into three categories based on the portion of the packet examined: the packet headers (this includes IP, TCP, UDP, and ICMP headers), specific bytes in the packet payload, or the entire payload of the packet.

Some Snort rules examine only the IP, TCP, UDP, or ICMP headers of packets. Any rules that don't use the Snort keywords `content`, `pcre`, `byte_test`, or `byte_jump`, examine only the packet headers. The following rule provides an example of a rule that examines only the TCP header.

```
alert tcp $EXTERNAL_NET 10101 -> $HOME_NET any (msg:"SCAN myscan";
        flow:stateless; ack:0; flags:S; ttl:>220;)
```

This rule looks for a TCP packet from an external address on port 10101 destined for any port on any local machine. The packet also must have the `ack` value set to 0, the `SYN` flag set, and a `TTL` greater than 220 for an alert to be generated.

Snort rules in the next category examine specific bytes within the packet. Rules that use the modifier keywords `depth`, `offset`, `within`, or `distance` following the `content` keyword or use the keywords `byte_test` or `byte_jump`, examine specific bytes in the packet payload. The `content` keyword defines a string that Snort will try to find. For example, the rule

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING speedera";
        itype:8; content:"89|3A 3B|<=>?"; depth:100;)
```

looks for an incoming ICMP echo request (`itype:8`) with the string ``89|3A 3B|<=>?'' in the first 100 bytes of the payload (`depth=100`). In addition to the keyword `depth` that limits how far into the packet to examine, the `offset` keyword defines how much of the packet to ignore before looking for the pattern. Snort also provides two keywords to limit what portion of the packet should be searched relative to the last content match. The keyword `within` defines how much of the packet to examine after the last pattern match. The keyword `distance` defines how much of the packet to skip before looking for the pattern. The keyword `byte_test` allows numerical comparison of specific bytes relative to the start of the packet or the last match. For example,

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"RPC network-status-monitor
mon-callback request UDP"; content:"|00 03 0D|p"; depth:4; offset:12;
content:"|00 00 00 01|"; within:4; distance:4; byte_test:2,<,200,0,relative;)
```

examines an incoming UDP packet and looks for a string in the 12^{th} to 16^{th} bytes of the packet. If the first string is found, the rule ignores the next 4 bytes (`distance=4`) looks for a string in the

subsequent 4 bytes (`within=4`) and checks whether or not the two bytes following the last matched string represent a number less than 200.

Snort rules typically examine the entire packet when trying to perform pattern matching. A `content` keyword without any additional keywords will force Snort to search the entire packet in order to find a match. In addition to the `content` keyword, Snort provides `pcre` to perform pattern matching using Perl-compatible regular expressions. The keyword `byte_jump` can force Snort to examine arbitrary bytes in the packet. The following rule examines the entire packet to search for the string `SSH-`. If `SSH-` is found, Snort converts the subsequent 4 bytes into a value, skips that many bytes, and uses a regular expression to check if there is not a carriage return in the next 200 bytes.

```
alert tcp $EXTERNAL_NET 22 -> $HOME_NET any (msg:"EXPLOIT SSH server banner
overflow"; content:"SSH-"; nocase;byte_jump:4,0,relative;
pcre:"/^SSH-\s[^\n]200/ism";)
```

The letters `ism` after the `/` define compile time flags for the regular expression. The suffix `i` indicates the rule is case insensitive, `s` tells the pattern-matching engine to include newline characters when matching the dot metacharacter, and `m` changes how anchors such as ^ and $ are handled. Without the `m` flag, ^ and $ limit the pattern matching to the beginning or end of the packet. When the `m` flag is included, Snort will try to anchor patterns to newlines. The addition of the `m` flag to rules with anchors forces Snort to search the entire string for possible matches, rather than just the first or last few bytes of the packet.

4. TAXONOMY OF SNORT RULES BY EVENT TYPE

To determine whether Two-Key IPsec could preserve Snort's capabilities, we analyzed the registered user release of "VRT Certified Rules for Snort CURRENT" dated 2006-05-24, consisting of 5089 rules. We found that roughly 3% of those Snort rules examine only the network-level and transport-level headers when detecting suspicious events. Thus, in order to determine whether there is any benefit to exposing the encrypted IP and transport header information using Two-Key IPsec, a deeper inspection of the rule set is necessary.

For the analysis presented in this report, we omit consideration of the network configuration. That is, we consider all possible rules, ignoring possible rule set reductions based on knowledge of available operating systems, available network services, and software versions. In addition, we don't consider rule dependencies, those cases where multiple rules are linked to detect a single suspicious event. For example, some Remote Procedure Call (RPC) attacks send a small number of benign packets prior to a final exploit packet. To lower false positives without introducing false negatives, Snort tracks both the benign and suspicious events through different rules and generates an alert only when every rule is matched. We treat these as largely independent rules. Since such event tracking is based on application-specific information associated with backdoors and trojans, treating the rules as independent doesn't impact the conclusions of our study.

The Snort rule set is subdivided into smaller, more manageable groupings. Most of the rule groupings are based on the vulnerable protocol or service, for example, SMTP, FTP, or ICMP. Some of the Snort rule groupings are based on the attack implementation, for example, *backdoor* or *virus*. Other rule groupings are general, for example, *misc*, *experimental*, and *local*. Finally, a subset of the Snort rule groupings is based on event type: *bad traffic*, *scan*, *DoS*, *DDoS*, *exploit*, and *policy violation*.

For our more detailed analysis, we wanted a categorization based exclusively on event type. To achieve that goal, we extended the event types defined by Snort as shown in Figure 2.

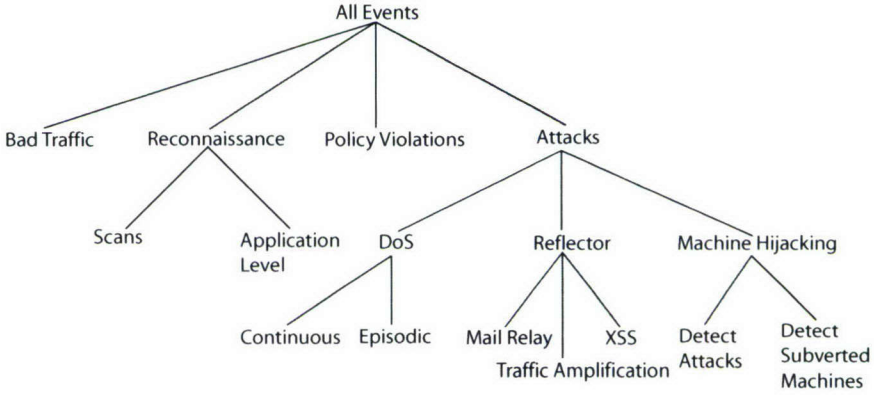


Figure 2. Taxonomy of Events of Interest to IDSs

There are four top-level categories in our taxonomy: bad traffic, reconnaissance, policy violation, and attacks. The taxonomy follows a tree structure in which children are more specific instances of their parents with the most specific event types at the leaves.

Bad traffic represents IP packets that should not traverse a network. Examples of bad traffic include invalid IP protocols; packets from the loopback address (127.0.0.1) or invalid addresses (e.g., 0.0.0.0); and BGP packets with invalid formats. Such packets could indicate malicious activities or misconfigured hosts.

Reconnaissance covers any type of information gathering. This category is subdivided by the technique used: **scans** use network-level and transport-level probes of IP addresses and ports while **application-level reconnaissance** utilizes knowledge of the application protocol. Scans can use ICMP ping requests or use UDP or TCP scanning tools (e.g., cybercop or nmap). On the other hand, application-level reconnaissance utilizes a wide range of tactics including querying mail servers for address lists, invalid URLs sent to web servers, DNS zone transfer requests, finger queries originating from outside of the network, and many other methods.

Policy violation consists of any types of activity which may not directly impact security, but are disallowed by a company. Snort has rules to detect peer-to-peer (P2P) or filesharing programs, the transfer of files, chat programs, and accessing inappropriate web content.

Attacks includes any action with a goal of subverting/hijacking a machine or disabling a machine or network. We subdivide attacks into three categories. Actions that can result in a machine or network being disabled are commonly referred to as **Denial of Service (DoS) attacks**. We further divide DoS attacks into **continuous DoS** and **episodic DoS** attacks. Continuous DoS attacks, such as network flooding attacks, require continued action to disable a service. Conversely, episodic DoS attacks, such as the "ping of death," use a small number of packets, perhaps as few as one, to shut down a service.

The second category of attacks is **reflector attacks**. A reflector attack exploits one network to attack another. We further divide reflector attacks into three categories: those where attackers use the exploited network as a **mail relay**, a **traffic amplifier** (e.g., a Smurf attack), or as a place to post malicious code (i.e., cross-site scripting (**XSS**) attacks). The defining feature of reflector attacks is that the host network may suffer no damage, excluding wasted resources.

The last category of attacks is the traditional **machine hijacking attack**. For Snort, we further divided this category into **detecting the attacks** and **detecting evidence of subverted machines**. Examples of this type of attack are buffer overflow exploits and format string exploits. Examples of evidence of machine subversion include active ports used by known backdoors or spyware and suspicious activities, such as password modification or root log ins, from hosts outside of the network.

5. ANALYSIS OF SNORT RULES

5.1 PORTION OF THE PACKET EXAMINED BY EVENT TYPE

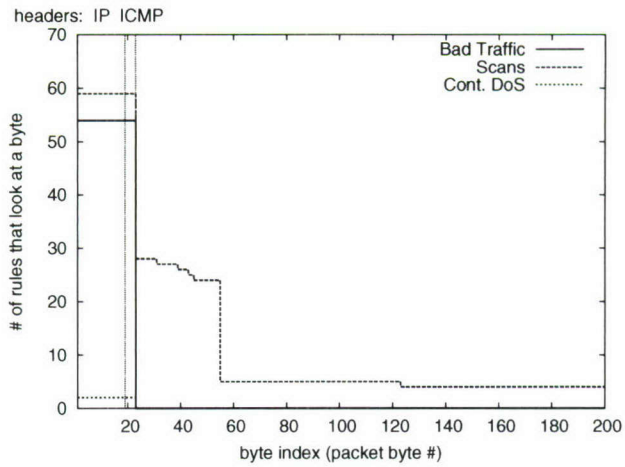
Table 1 indicates the total number of rules and the number and percentage of rules that examine the entire packet and that examine only the packet headers, for each event type in our taxonomy. Fifty of the Snort rules are excluded from the results in the table since they are used to track the initiation status for TLS/SSL connections. Those rules do not generate alerts and are used to set flags that are subsequently checked by multiple rules in the episodic DoS and detect hijack attack categories. The majority of rules that detect application-level reconnaissance, episodic DoS, reflector attacks, machine hijack attacks, and subverted machines need access to the entire packet to accurately detect the various events. However, fewer than 10% of the rules for detecting bad traffic, scans, and continuous DoS events examine the entire packet. In addition, Snort may be configured to use a preprocessor to detect a larger range of scans. This preprocessor uses only information in the IP and TCP/UDP headers to detect scans. These are positive results, given that bad traffic and scans are network-level events that HIDSs may miss. This analysis indicates that Snort, and by extension other NIDSs, can provide a necessary service without forcing clients to expose the entire contents of the packet to a network service.

Event Type	# of Rules	#(%) Examine Entire Packet	#(%) Examine Only Headers
Bad Traffic	80	7(8.8%)	70(88%)
Scans	83	8(9.6%)	42(50%)
Application Reconn.	464	368(79%)	13(2.8%)
Continuous DoS	5	0(0%)	2(40%)
Episodic DoS	132	104(79%)	14(11%)
Reflector Attacks	21	20(95%)	0(0%)
Detect Hijack Attack	3400	3251(96%)	63(1.8%)
Subverted Machines	742	630(85%)	9(1.2%)
Policy Violation	112	72(64%)	4(3.5%)

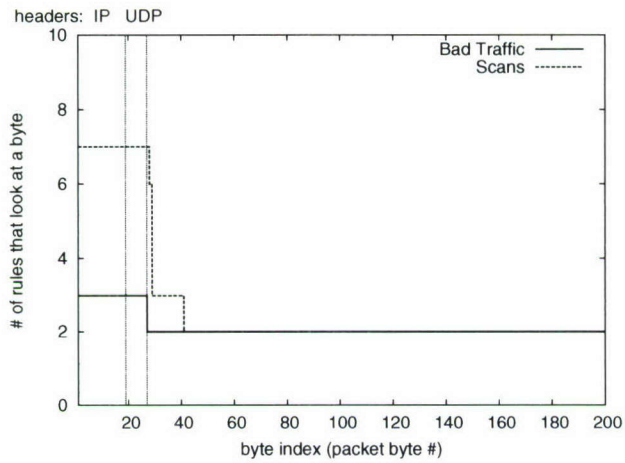
TABLE 1. Taxonomy Statistics

While Table 1 shows that the majority of bad traffic, scans, and continuous DoS events are detectable without examining the entire packet, it does not convey how little of the packet actually needs to be exposed. Figure 3 shows histograms of the number of bad traffic, scan, and continuous DoS rules that examine each byte of the IP packet. Each histogram shows the results for Snort rules based on a specific IP protocol: ICMP, UDP, or TCP. The 9 bad traffic and 2 scan rules that use IP headers only are included in the counts in Table 1 but not in the histograms. In the histograms, only byte indices up to 200 are shown. For these plots, rules that examine more than 200 bytes examine the entire packet.

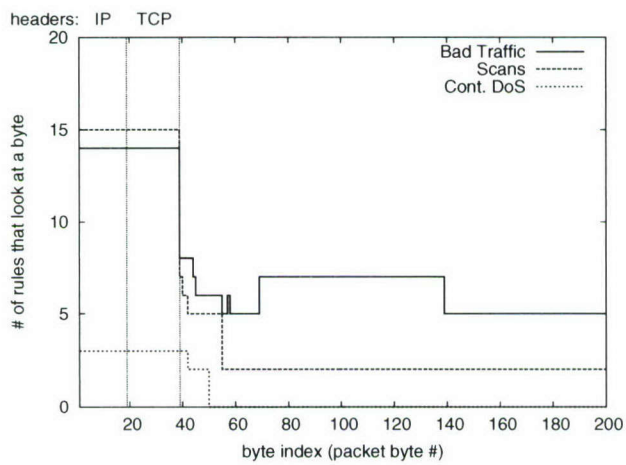
For the plots in Figure 3 we assume there are no IP or TCP options, i.e., the IP header is



a) Histogram of the 54 Bad Traffic, 59 Scan, and 2 Cont. DoS ICMP Rules



b) Histogram of the 3 Bad Traffic and 7 Scan UDP Rules



c) Histogram of the 14 Bad Traffic, 15 Scan, and 3 Cont. DoS TCP Rules

Figure 3. Histogram of How Many Rules in Each of 3 Categories Examine Each Byte in a Packet

from byte 0 to byte 19 and the TCP header is from bytes 20 to 39. Each plot has vertical lines to define where one header ends and the next header or the packet's contents begin. For example, in Figure 3a the solid line representing bad traffic indicates 54 rules examine bytes 0 through 23 (the IP and ICMP headers), and 0 rules examine any bytes past the ICMP header (a value of 0 for any byte index > 23).

A network designer could configure Two-Key IPsec to encrypt only the IP and ICMP/UDP/TCP headers of packets using the shared key. In that case, the 217 Snort rules included in the fourth column of Table 1 and depicted as using only header information in the histograms in Figure 3 could be used by a NIDS with access to the shared key. In addition, the histograms show that the majority of the remaining bad traffic, scan, and continuous DoS rules examine fewer than 60 bytes at the beginning of the packet. For example, the ICMP rule

```
alert icmp $EXTERNAL_NET any → $HOME_NET any (msg:"ICMP PING Microsoft Windows";
itype:8; content:"0123456789abcdefghijklmnop"; depth:32; reference:arachnids,159;
classtype:misc-activity; sid:376; rev:7;)
```

examines only the first 56 bytes of an ICMP packet (24 bytes for the IP and ICMP headers and 32 bytes of the ICMP payload) to detect an IP address scan. First, the rule checks that the packet is an ICMP echo request (`itype:8`), commonly referred to as a ping request. It then categorizes the originator as a machine running Microsoft Windows by examining up to 32 bytes (`depth:32`) of content for the string `"0123456789abcdefghijklmnop"`.

Of the 38 ICMP rules that examine the packet contents, all are scan rules and all but one are checking ICMP echo request and reply messages. That one rule checks the first 22 bytes of the ICMP payload, of all ICMP packets originating outside the local network, for a Digital Island bandwidth query. A HIDS can detect this event and prevent the host from responding. It may also be possible to develop a rule that checks for ICMP request messages.

For the remaining 37 ICMP scan rules, we observe that ICMP echo request and reply messages should not be carrying sensitive information. In that case, Two-Key IPsec can be used to encrypt the full ICMP payload, as well as the header information, with the shared key. The NIDS will then have full access to the ICMP packet and will retain its original fidelity and false-alarm rate.

Even if Two-Key IPsec is used to encrypt only the first 56 bytes of all ICMP echo request and reply messages using the shared key, all but six of the rules in Figure 3a can be used by the NIDS. Further, if decrypting the ICMP payload is unacceptable, then the NIDS could use a single rule to detect ICMP echo request and reply messages, instead of using 37 separate rules. Although the fidelity may be lower and false alarms greater, such a rule would require use of only IP and ICMP header information and would give a NIDS using Two-Key IPsec nearly full coverage of ICMP rules for bad traffic, scans, and continuous DoS events.

Coverage by the TCP and UDP rule sets could also be improved if additional bytes in the packets are made available to a NIDS using a shared key. Unfortunately, since UDP and TCP payloads are likely to contain information that should remain confidential, it is not practical to enable use of those types of rules using the Two-Key IPsec approach. We examine whether these TCP and UDP rules can be rewritten to allow the NIDS to operate later in this section.

Even if Two-Key IPsec were to be used to give the NIDS access to the beginning of UDP and TCP packets and full access to ICMP ping packets, 7 bad traffic and 4 scan rules that examine the entire packet remain. Of those bad traffic rules, two detect spoofed DNS responses, one detects invalid BGP messages, two detect malformed NetBIOS requests, one detects a malformed rlogin message, and one detects a NIDS evasion technique used when attacking web servers. Of the four scan rules, one looks for Distributed DoS (DDoS) communication between a client and its master hosts, another looks for a UDP probe containing a specific string, and two look for trojan activity. A HIDS, particularly one that operates at or below the transport layer, can detect all of these events and protect targeted applications. In the case of the DDoS communication and trojan activity, if the host has been compromised, there is a strong chance the HIDS has been disabled. In the next subsection, we look at whether these three DDoS and trojan activity detection rules can be rewritten to allow detection by a NIDS.

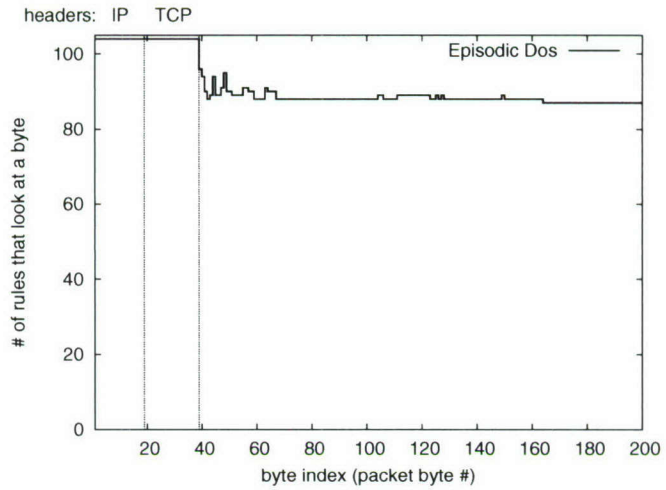


Figure 4. Histogram of the 104 Episodic DoS TCP Rules

In contrast to Figures 3a-c, Figure 4 shows the bytes examined by the episodic DoS rules for TCP traffic. For a NIDS to detect the majority of episodic DoS activities, a large portion of the packet must be exposed. Thus, most episodic DoS attacks are best detected by a HIDS.

We next look at whether it is possible to increase the effectiveness of the NIDS by rewriting some of the rules to require access to less of the packet.

5.2 WHETHER SIGNATURES NEED TO SEARCH THE ENTIRE PACKET

We showed in Table 1 that Snort searches the entire packet when detecting the majority of application-level reconnaissance, episodic DoS, reflector attacks, machine hijack attacks, subverted machines, and policy violation events. However, it was not clear whether Snort rules must search the entire packet. We wanted to determine whether designers could rewrite the Snort rules to

examine less of the packet and thereby make the rules usable by a NIDS using Two-Key IPsec. We examined the rules with two specific goals. The first goal was to increase the number of UDP and TCP rules available to the NIDS for detecting bad traffic, scan, and continuous DoS events. The second goal was to make the large number of rules for detecting machine hijack attacks targeting web, mail, and Remote Procedure Call (RPC) servers available to the NIDS.

To accomplish the first goal, we examined the characteristics of the 9 UDP rules and 20 TCP rules that access the packet payload. Of those, 12 rules can be attributed to network or host compromise, 4 to network reconnaissance with no compromise, and the remaining 13 to attacks against various application services. A HIDS, particularly one like Snort that operates below the network layer, is well-suited to protecting application services so we focused our efforts on the 16 rules we would like a NIDS to be able to implement.

The rule used to detect a DDoS client searching for its master is

```
alert udp $EXTERNAL_NET any → $HOME_NET 31335 (msg:"DDOS Trin00 Daemon to Master
PONG message detected"; content:"PONG"; reference:arachnids,187;
classtype:attempted-recon; sid:223; rev:3;)
```

This rule is looking for a UDP packet destined for port `31335` on a local host and containing the string `"PONG"` anywhere in the payload. Although this rule cannot be rewritten to provide the same functionality without accessing the payload, a rule could be written for the NIDS to report on suspicious activity on UDP port `31335`. Since this port is widely known to be used by Trin00, it should be blocked in the network. Thus, providing the NIDS with access to the IP and UDP headers will allow it to detect and respond to evidence of active Trin00 DDoS software.

We did a similar analysis of the other 15 rules. Of those rules, 10 use a characteristic port that could be used for detection at the risk of increasing the rate of false alarms. The false-alarm rate could potentially be reduced by adding checks for packet size, requiring repeated detections for alerting, and adding checks for preconditions. However, we leave analysis of the false-alarm rate for future work.

Three of the remaining five rules are used for detecting Cybercop scans. For example,

```
alert tcp $EXTERNAL_NET any → $HOME_NET any (msg:"SCAN cybercop os PA12 attempt";
flow:stateless; flags:PA12; content:"AAAAAAAAAAAAAAAA"; depth:16; sid:626;
rev:8;)
```

examines the first 56 bytes of a TCP packet (40 bytes for the IP and TCP headers plus 16 bytes of payload) to detect a scan. First the rule checks if the `Push`, `Ack`, and first and second reserved bits (`PA12`) of the packet are set. If those bits are set, the additional 16 bytes are examined to identify the scanning tool. The other two Cybercop scan rules are similar except one checks for `flags:SFU12` (`Synchronize`, `Finish`, `Urgent`, and the two reserved bits are set) and the other checks for `flags:SFP`. Cybercop uses these flags to fingerprint the OS of the target machine. Since the content check is used for identifying the scan tool, that check can be omitted. A higher false-alarm rate is only likely to occur if Explicit Congestion Notification (ECN) is enabled, since ECN uses the first and second reserved bits. In that case, the `flags:PA12` rule should be removed from the active rule set.

The remaining two rules rely exclusively on the packet payload contents for detection and cannot be rewritten and still remain effective. The first is a scan rule that examines the entire packet to detect an attempt to contact the FsSniffer trojan running on a compromised host:

```
alert tcp $HOME_NET any → $EXTERNAL_NET any (msg:"BACKDOOR FsSniffer connection
attempt"; flow:to_server,established; content:"RemoteNC Control Password|3A|";
reference:nessus,11854; classtype:trojan-activity; sid:2271; rev:2;)
```

Since this rule cannot be rewritten to be usable by a NIDS using Two-Key IPsec, it must be used by a HIDS. Since the rule is checking for an outgoing connection, the HIDS can effectively detect the event, unless the HIDS has been disabled.

The second rule that relies exclusively on the packet payload for detection is a bad traffic rule. It examines the first 46 bytes of the packet (the IP and TCP headers plus 6 bytes of the TCP payload) to determine if the SecureNetPro IDS is running on the network:

```
alert tcp $EXTERNAL_NET any → $HOME_NET any (msg:"OTHER-IDS SecureNetPro
traffic"; flow:established; content:"|00|g|00 01 00 03|"; depth:6;
classtype:bad-unknown; sid:1629; rev:6;)
```

This rule detects 16 bits of `0`'s followed by `"g"` followed by the integer value `1` and the integer value `3` in the packet payload. This rule cannot be rewritten to be usable by a NIDS using Two-Key IPsec. It is, therefore, effective only if used by a HIDS running on the host running SecureNetPro. As with the previous rule, the HIDS can effectively detect the event, unless the HIDS has been disabled.

We have established that, by rewriting some of the bad traffic and scan rules, we are able to achieve our first goal and increase the number of rules available for use by a NIDS using Two-Key IPsec, albeit with more false alarms. In addition, we have established that rules that cannot be used by the NIDS can be implemented in the HIDS with no loss of coverage.

Our second goal was to enable the NIDS to use a more significant percentage of the large number of rules for detecting machine hijack attacks. We examined several servers to determine if attackers can move exploit-identifying data to arbitrary positions in the packet. Our testing found that popular web and mail servers will accept a wide range of inputs that permit attackers to move exploit code to an arbitrary position in the packet. We found Apache (v1.3.31, v2.0.52, and v2.0.54) and Microsoft Internet Information Server (IIS) (v5 and v6) web servers permit an arbitrary number of spaces between an HTTP method (e.g., `GET`) and the associated filename (e.g., `index.html`) or the use of arbitrarily long relative paths (e.g., `./././././`). We also found that SendMail (v8.11.7 and v8.12.10), PostFix (v10.3), and Exchange (v5 and v6) are not strict about accepting suspicious inputs. Both PostFix and SendMail permit an arbitrary number of spaces before commands (e.g., `HELO`, `FROM`, etc.). All three mail servers permit multiple commands in a single packet, allowing attackers to send benign commands prior to malicious ones, and arbitrary spaces between commands and parameters, allowing attackers to simply move the malicious parameters deeper into the packet. Thus, rules associated with these services must be written to examine the entire packet.

Our analysis of the Microsoft Windows Distributed COM (DCOM) RPC service shows it is much stricter than the mail and web servers. The DCOM RPC service accepts input only in

a specific format, even before the vulnerability associated with the Blaster [25] and Welchia [26] worms is patched. The DCOM RPC service rejects any packet in which header fields have been manipulated in order to move parameters deeper into the packet. Thus, designers could rewrite rules related to the Windows DCOM RPC vulnerability to examine less of the packet since the malicious data is limited to a specific region.

In general, the vulnerable service dictates whether or not an attacker can move attack code within the packet. As long as some services permit the movement of attack code, Snort rules associated with those services must search the entire packet and the IDS must access the entire packet. Thus, HIDS must be used to detect machine hijack attempts targeted at servers.

6. PROPOSED SOLUTION

Based on the results of the previous section, it is clear that if normal IPsec is used, a NIDS will have access to only IP and ESP packet headers. Only 39 of the Snort rules will function with such little information (e.g., bad traffic rules that detect invalid addresses such as `0.0.0.0` or `127.0.0.1`). Further, we also showed that to detect every event, Snort must search the entire packet. Even with router-based flow monitoring based on IP addresses and network interface number, it may not be possible to distinguish traffic associated with a continuous DoS attack from other traffic with the same source IP address crossing the same router interface.

One solution is to dispense with a NIDS and rely exclusively on HIDSs. However, some network-level events could be hard to detect using HIDSs alone, hosts under continuous DoS attacks might be too overloaded to generate alerts and HIDSs can be subverted if hosts are compromised.

Another solution is to expose the entire packet to the NIDS, as is currently done in commercial security gateway products. If an attacker were to compromise the NIDS in such a gateway (as with the Witty worm [22]), IPsec would provide no protection from eavesdropping, forging, or modification of any traffic destined to or originating from **any host** in the network using IPsec.

A third solution, and the one we propose, uses a hybrid approach. In the previous section, we demonstrated that a NIDS can detect certain network-level events when the first few bytes of the packet are exposed. In order to increase the number of rules that could be used by a NIDS, we rewrote some of the rules and proposed allowing the NIDS full access to ICMP echo request and reply messages. We showed that with some changes to the rules and using HIDSs, a NIDS, and Two-Key IPsec, traffic can be protected and all events can be detected, albeit with a possible increase in the false-alarm rate. If one of these three components is missing, the defenses may not be able to detect some events in the taxonomy (e.g., Snort deployed with normal IPsec cannot detect most scans, a HIDS may not detect misconfigurations that generate bad traffic, and without a HIDS, Snort and Two-Key IPsec cannot detect attempts to hijack machines). Figure 5 indicates what type of IDS covers each part of the event taxonomy. When used with Two-Key IPsec, a NIDS can detect most bad traffic, scans, and continuous DoS events. A HIDS can detect the remainder of the events in the taxonomy, but may have trouble generating timely alerts for scans, bad traffic, or continuous DoS.

A HIDS has access to data after the host decrypts the packet and can achieve a better understanding of system state than a network-based system. With full access to traffic, a HIDS can detect the type of events that force Snort rules to search the entire contents of the packet (e.g., application-level reconnaissance, machine hijack attempts, or policy violations).

A NIDS, in conjunction with Two-Key IPsec, provides services a HIDS cannot perform and an added level of protection through defense-in-depth. As discussed in Section 5, a NIDS can detect scans and bad traffic while examining a small portion of the packet. If scans are destined for unassigned addresses or misconfigured machines generate bad traffic, a HIDS may never receive the traffic, but a NIDS monitors all traffic across the network and can detect such events. Even with only a small portion of the packet exposed, the NIDS can provide some defense in depth. If a machine is compromised, a NIDS may not detect events that indicate an attacker has subverted

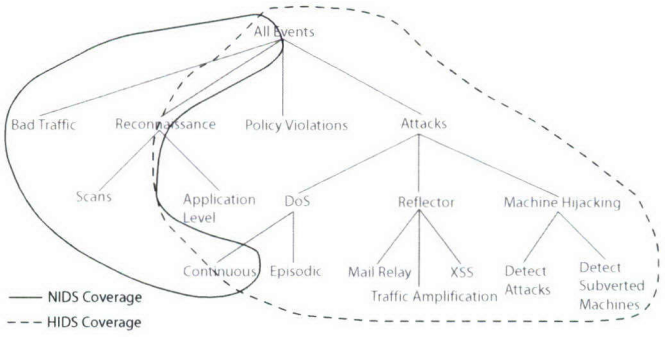


Figure 5. NIDS & HIDS Coverage of the Taxonomy

that machine (e.g., the installation of a root kit or some backdoor). However, the NIDS can detect if the attacker uses the subverted machine to scan other local hosts or exfiltrate data. If the attacker disables the HIDS, the NIDS alert is very important since administrators may not receive any other alerts.

Under Two-Key IPsec, an attacker who compromises the NIDS can **read only** the IP header, ICMP, TCP, or UDP header, and the small portion of the packet encrypted under the shared key. Note that the ability of the attacker to modify or create new packets is dependent on how the encrypted portions of the packet are authenticated. If the encrypted portions of the packet are authenticated from end to end, it will not be possible for the attacker to successfully modify the packet or create new packets. If the separately encrypted portions of the packet are authenticated separately, it may be possible for the attacker to modify a portion of the packet, or even create new packets without being detected.

7. DISCUSSION

A conventional assumption is that NIDSs will be rendered largely ineffective by IPsec unless the packet contents can be decrypted by the NIDSs. We propose an alternative safer solution that uses HIDSs, a NIDS, and Two-Key IPsec to improve the NIDS's effectiveness with IPsec-encrypted traffic.

While Two-Key IPsec provides additional functionality over standard IPsec, it also introduces additional complexity and overhead. Layered Encryption Security (LES) [11] minimizes the computational overhead by using a fast cipher, RC5, for encrypting the header information. Except for an additional encryption over the header, the rest of the IPsec processing remains the same. LES reduces the added network overhead by constraining the number of encryption zones and the locations of those zones. Sing and Soh [23] address some of the complexity and overhead issues with Multilayer IPsec (ML-IPsec) [33]. They limit the number of encryption zones to three and define those zones based on semantic information. By limiting the zones in that way, the network overhead can be reduced. Further, since they define the zones as contiguous bytes of information, computational overhead is reduced as well. That reduction may be offset, however, by the computation required for parsing the packet content.

One issue with all the proposed Two-Key IPsec solutions is that a key distribution mechanism is needed, beyond what is provided using IKE, to distribute keying material for the additional encryption zones. The implementations of LES and ML-IPsec sidestep this issue by using preplaced symmetric keys for encryption zones that are accessed by intermediate network services.

Beyond the complexities of the IPsec implementation, many NIDSs, Snort among them, receive packets directly from the network interface before they are processed by IPsec. Thus, the NIDS will need to either use a preprocessor for decryption or be able to decrypt the information itself. Assuming the infrastructure is provided to allow the NIDS to utilize the IPsec processing mechanisms, the additional complexity of the NIDS should be minimal. Another issue for the NIDS is additional delay caused by decryption. Any processing delay associated with retrieving the appropriate key and performing decryption could have an impact on the ability of the NIDS to keep up with the input data rate. If the NIDS must drop or queue packets to keep up then delayed warnings or false negatives are likely. However, decryption times are typically small and only a small portion of the packet will be decrypted using Two-Key IPsec. Further, there is ongoing research [6, 7, 15, 16] to improve the performance of NIDSs and deal with growing traffic rates.

Provided a NIDS without IPsec can keep up with the maximum traffic rates, a system with Two-Key IPsec should not have much trouble. If the network is small, the NIDS can store information associated with the various keys in a hash table or some other quickly accessed structure. For large networks, administrators could divide the network into subnetworks with a NIDS for each.

A final issue is that although NIDSs are known for generating large numbers of alerts, many of them false, some of the changes we suggest making to the rules may exacerbate the problem. On the other hand, tailoring the rule set to the network topology and configuration, coordinating the NIDS with HIDSs, using attack graphs [8] to filter alerts, and incorporating behavioral analysis of network traffic could lower the false-alarm rate.

8. CONCLUSIONS AND FUTURE WORK

System administrators commonly use NIDSs to protect their networks. A common concern is that more widespread use of IPsec for encrypting network traffic will render NIDSs virtually useless. Common solutions rely on decrypting the entire packet at the NIDS or placing IDSs on the end hosts. The first technique can have a negative impact on privacy while the second technique can have a negative impact on security.

Another solution relies on the use of multiple encryption zones. Under this approach, the packet headers are encrypted with one key and the packet payload is encrypted with a different key, known to only the endpoints. We refer to this type of encryption as Two-Key IPsec. For this study, we looked at the efficacy of using Two-Key IPsec in conjunction with Snort to detect suspicious network traffic.

We found that when Snort rules were categorized by event type, it was possible to identify rules that will detect network-level events when a portion of the packet is made available to the NIDS. Thus, using Two-Key IPsec to expose the IP packet header information will allow a NIDS, such as Snort, to detect the majority of bad traffic, scans, and continuous DoS events. Such events cannot be detected by a NIDS when the IP packet is encrypted using regular IPsec and cannot be effectively detected by HIDSs. We also found that it is possible to increase the number of bad traffic, scan, and continuous DoS events that can be detected by a NIDS using Two-Key IPsec. We did this by rewriting the rules, or in the case of ICMP echo request and reply messages, allowing the NIDS access to the ICMP payload. However, we found that for machine hijack attacks against servers, in many cases it is not possible to reduce the number of Snort rules that examine the entire packet since some vulnerable services permit the movement of attack code in the user input. Even when rules can be rewritten to allow a NIDS to operate, there may be a resulting increase in the false-alarm rate. We are pursuing further research into ways to reduce the false-alarm rate.

Thus, to detect all events detectable on unencrypted traffic, while protecting packet contents from eavesdroppers, a combination of HIDSs, a NIDS, and Two-Key IPsec must be employed. Although Two-Key IPsec will allow Snort to operate in networks containing encrypted traffic, there are several barriers to its use. First, the use of encryption zones as a feature of IPsec is not widely accepted. Second, although there are several prototype implementations, all have shortcomings that need to be addressed. Finally, use of the information made available to a NIDS by Two-Key IPsec needs further examination. Our current research is exploring additional approaches to alleviate these shortcomings.

For future work, if more of the application header of the packet can be exposed, then more of the Snort rules would be able to evaluate encrypted traffic. There are two thrusts that could be taken along these lines. One is something we call Intelligent Two-Key IPsec, which would support using a separate key to encrypt the application headers. Another thrust is to examine how the application headers could be exposed for TLS.

GLOSSARY

BGP	Border Gateway Protocol
DCOM	Distributed COM for Microsoft Windows
DNS	Domain Name Service
DDoS	Distributed Denial of Service
DoS	Denial of Service
ESP	Encapsulating Security Payload
FTP	File Transfer Protocol
HIDS	Host-based Intrusion Detection System
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPsec	Internet Protocol Security
LES	Layered Encryption Security [11]
ML-IPsec	Multi-Layer IPsec [32]
NIDS	Network-based Intrusion Detection System
P2P	Peer-to-Peer
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer (see also TLS)
TCP	Transmission Control Protocol
TLS	Transport Layer Security (see also SSL)
UDP	User Datagram Protocol
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

REFERENCES

[1] IP Encapsulating Security Variable Payload (ESVP). IETF Internet Draft, IETF, `http://tools.ietf.org/id/draft-kasera-esvp-00`, 2002.

[2] William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin. *Firewalls and Internet Security, Second Edition: Repelling the Wily Hacker*, pages 10, 281. Addison-Wesley, 2003.

[3] Cisco Systems, Inc. Cisco PIX firewall IPsec user guide. `http://www.cisco.com/en/US/products/sw/secursw/ps2120/products_user_guide_book09186a008008c38f.html`, 2006.

[4] Cisco Systems, Inc. Cisco IOS security configuration guide (Releases 12.4 T), Cisco Group Encrypted Transport VPN. `http://www.cisco.com/en/US/products/ps6441/products_feature_guide09186a008078e4f9.html`, 2007.

[5] Dorothy E. Denning. The Clipper encryption system. *American Scientist*, 81(4):319–323, July 1993.

[6] Holger Dreger, Anja Feldmann, Vern Paxson, and Robin Sommer. Operational experiences with high-volume network intrusion detection. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, 2004.

[7] Jose M. Gonzalez and Vern Paxson. Enhancing network intrusion detection with integrated sampling and filtering. In *Proceedings of the 9th International Symposium On Recent Advances In Intrusion Detection (RAID 2006)*, Hamburg, Germany, September 2006.

[8] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference, 2006 (ACSAC '06)*, pages 121–130, December 2006.

[9] Juniper Networks, Inc. Integrated firewall VPN security. `http://www.juniper.net/products/integrated/`, 2006.

[10] Manish Karir. IPSEC and the Internet. Master's Dissertation, University of Maryland, Department of Electrical Engineering, 1999.

[11] Manish Karir and John Baras. LES: Layered Encryption Security. In *Third International Conference on Networking (ICN'04)*, 2004.

[12] Sneha Kumar Kasera, Semyon Mizikovsky, Ganapathy S. Sundaram, and Thomas Y. C. Woo. On securely enabling intermediary-based services and performance enhancements for wireless mobile users. In *Workshop on Wireless Security*, pages 61–68, 2003.

[13] S. Kent. IP Encapsulating Security Payload (ESP). RFC 4303, Internet Engineering Task Force, December 2005.

[14] S. Kent and K. Seo. Security architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005.

[15] C. Kruegel, F. Valeur, G. Vigna, and R.A. Kemmerer. Stateful Intrusion Detection for High-Speed Networks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 285–293, Oakland, CA, May 2002. IEEE Press.

[16] Wenke Lee, Joao B. D. Cabrera, Ashley Thomas, Niranjan Balwalli, Sunmeet Saluja, and Yi Zhang. Performance adaptation in real-time intrusion detection systems. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, 2002.

[17] Cynthia McLain, Roger Khazan, and Siddhartha Maddi. Securely managing encrypted IP communications. Work in progress, 2007.

[18] Phillip Porras. Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD). `http://www.sdl.sri.com/projects/emerald/`.

[19] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proceedings of the USENIX 13th Systems Administration Conference (LISA '99)*, pages 229–238, November 1999.

[20] A. Roy-Chowdhury, J. S. Baras, M. Hadjitheodosiou, and S. Papademetriou. Security issues in hybrid networks with a satellite component. *IEEE Wireless Communications*, 12(6):50–61, 2005.

[21] M. M. Shahsavari and N. Z. Almeshary. Enabling the intelligent network services in the presence of the end-to-end security model of windows XP/2000's IPSec protocols using two layer protection model. In *Seventh IASTED International Conference on Internet and Multimedia Systems and Applications*, 2003.

[22] Colleen Shannon and David Moore. The spread of the witty worm. `http://www.caida.org/analysis/security/witty/`, 2006.

[23] Joel Sing and Ben Soh. A critical analysis of multilayer IP security protocol. In *ICITA '05: Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2*, pages 683–688, Washington, DC, USA, 2005. IEEE Computer Society.

[24] Symantec. Perimeter to the core protection. `http://enterprisesecurity.symantec.com/article.cfm?articleid=3125`, 2003.

[25] Symantec. W32.Blaster.Worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html`, 2003.

[26] Symantec. W32.Welchia.Worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html`, 2003.

[27] Symantec. Integrated protection and todays security threats. `http://enterprisesecurity.symantec.com/article.cfm?articleid=5134`, 2004.

[28] Symantec. Symantec Gateway Security 5600 series. http://www.symantec.com/enterprise/products/overview.jsp?pcid=1001&pvid=941_1, 2006.

[29] Symantec. Understanding comprehensive threat management. http://ses.symantec.com/article.cfm?articleid=6585&EID=0, 2006.

[30] Symantec. Symantec Sygate Enterprise Protection. http://scan.sygate.com/enterprise/products/overview.jsp?pcid=1322&pvid=1303_1, 2007.

[31] Yongguang Zhang. Multi-layer protection scheme for IPsec. IETF Internet Draft, IETF, http://tools.ietf.org/html/draft-zhang-ipsec-mlipsec-00, 1999.

[32] Yongguang Zhang. A Multi-Layer IP security protocol for TCP performance enhancement in wireless networks. *IEEE Journal on Selected Areas in Communications*, 22(4):767–776, May 2004.

[33] Yongguang Zhang and Bikramjit Singh. A Multi-Layer IPsec protocol. In 9^{th} *USENIX Security Symposium*, pages 213–228, August 2000.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED *(From - To)*
11 May 2007	Technical	

4. TITLE AND SUBTITLE
Making Network Intrusion Detection Work with IPsec

5a. CONTRACT NUMBER
FA8721-05-C-0002

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
1385

6. AUTHOR(S)
C.D. McLain, A. Studer, R.P. Lippmann

5d. PROJECT NUMBER

5e. TASK NUMBER
331

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420-9108

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-1121

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Air Force ESC/XPKL
5 Eglin Street
Hanscom AFB, MA 01731

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ESC-TR-2006-080

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Network-based intrusion detection systems (NIDSs) are one component of a comprehensive network security solution. The use of IPsec, which encrypts network traffic, renders network intrusion detection virtually useless unless traffic is decrypted at network gateways. One alternative to NIDSs, host-based intrusion detection systems (HIDSs), provides some of the functionality of NIDSs but with limitations. HIDSs cannot perform a network-wide analysis and can be subverted if a host is compromised. We propose an approach to intrusion detection that combines HIDS, NIDS, and a version of IPsec that encrypts the header and the body of IP packets separately. We refer to the latter generically as Two-Key IPsec. We show that all of the network events currently detectable by the Snort NIDS on unencrypted network traffic are also detectable on encrypted network traffic using this approach. The NIDS detects network-level events that HIDSs have trouble detecting and HIDSs detect application-level events that can't be detected by the NIDS.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	Same as report	39	